



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/723,109	11/25/2003	Ian Nicholas Whalley	YOR920030421US1	1282
29683	7590	06/20/2006	EXAMINER	
HARRINGTON & SMITH, LLP 4 RESEARCH DRIVE SHELTON, CT 06484-6212			PHAM, THAI V	
			ART UNIT	PAPER NUMBER
			2194	

DATE MAILED: 06/20/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 10/723,109	Applicant(s) WHALLEY, IAN NICHOLAS	
	Examiner Thai Van Pham	Art Unit 2194	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 11/25/2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-36 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-36 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 25 November 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

This is the initial office action based on the application filed on June 9, 2006. Claims 1 – 36 are currently pending and have been considered below.

Claim Objections

1. Claims 33 and 34 are objected to because of the following informalities: containing acronyms that are not explicitly spelled out in their parent claims although are explained in the specification.

-- Claim 33: TSD/FSFD.

-- Claim 34: LD_LIBRARY_PRELOAD

Appropriate correction is required.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

2. The following claims contain insufficient antecedent bases for the limitations in the claims.

-- Claims 8, 17 and 26: recite the limitation "the data objects". The Examiner assumes that "data objects" is meant here as the context of the claim is taken into consideration.

-- Claims 9, 18 and 27: recites the limitation "the project building tool". The Examiner assumes that "the software project building tool" is meant here as the context of the claim is taken into consideration.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

3. Claims 1 – 6, 10 – 15, and 19 – 24 are rejected under 35 U.S.C. 102(e) as being anticipated by **Willadsen (10/281,945)**.

-- Claims 1, 10, and 19: **Willadsen** discloses an apparatus and software management method – which can be directed by a computer program stored on a computer readable medium – for performing software configuration management operations and for supporting build processes. The apparatus comprises:

a) a processor which is capable of executing all and any software applications or modules (e.g., data transformation tool) that are specific to the operating system on which the processor runs (Fig. 1, [0046]);

b) a change management monitoring and decision sub-modules which monitor software components to detect changes in the software components as well as explicit

and/or implicit dependencies (i.e., monitoring the operation of the data transformation tool) (Figs. 5 – 6, [0058 – 0061]);

c) an initial management information data generator and change management control modules automatically determine explicit and implicit dependencies for the software components (Figs. 3 – 4, [0053 – 0055]).

-- Claims 2 – 3, 11 – 12, and 20 – 21: In the system of Claim 1, **Willadsen** further discloses that the change information of software components may include information indicating the date, time, and revision number of a software component and/or application. If a change is detected, the detected change information is processed to generate change information of its explicit/implicit dependencies. The change information of dependencies is then stored in memory (Fig. 5, [0058]).

-- Claims 4, 13, and 22: In the system of Claim 1, **Willadsen** further discloses that the monitored software component can be a program component, data component or combined program-data component (i.e., files in a file system) ([0035]). Furthermore, the dependency relationship information generated by the system can be propagated and inherited within or across projects in a software control management tool (i.e., software project building tool) ([0015 – 0020]).

-- Claims 5, 14, and 23: In the system of Claim 4, **Willadsen** further discloses that the dependency relationship information generated by the system can be

Art Unit: 2191

propagated and inherited within or across projects in a software control management tool ([0017, 0035]). Software control management tools, such as VCS, Perforce, and ClearCase, are inherently capable of monitoring and recording changes in files as well as attributes of a version of a project. These changes include but not only limited to creation and modification as well as explicit and implicit dependencies of files. Furthermore, the change management monitoring and decision sub-modules in the system of Claim 1 of **Willadsen** monitor these changes to determine and update explicit/implicit dependency information of a project files (Figs. 5 – 9).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

4. Claims 6 – 8, 15 – 17, and 24 – 26 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Willadsen (10/281,945)**.

-- Claims 6, 15, and 24: **Willadsen** discloses a system as in Claim 4 above which is incorporated in software control management tools; however, he does not explicitly mention a software control management tool (i.e., software project building tool) may contain a plurality of sub-tools. Nevertheless, Official Notice is taken that it is old and well known in the art of software development that software version control tools, such as ClearCase, contain not only different modules (i.e., sub-tools) that are capable of

Art Unit: 2191

monitoring and recording changes of files and attributes in a project version, but also different target compilers (i.e., subtools) where each compiler contains a parser, a syntax analyzer and translator, an object linker, a machine code generator, etc. Thus, it would have been obvious to one with ordinary skills in the art of software development at the time the invention was made to recognize that the system disclosed by **Willadsen** could be employed in applications (e.g., project version control) containing one or more sub-tools.

In the system of Claim 4, **Willadsen** further discloses that the change management monitoring and decision sub-modules monitor software components to detect changes in the software components as well as explicit and/or implicit dependencies. The change information is then organized in a Nodal Tree Structure and recorded in memory (i.e., in table form) (Figs. 5 – 6, [0054, 0058 – 0061]). Furthermore, **Willadsen** discloses that the dependency relationship information generated by the system can be propagated and inherited within or across projects in a software control management tool ([0017, 0035]). Software control management tools, such as VCS, Perforce, and ClearCase, are inherently capable of monitoring and recording changes in files as well as attributes of a version of a project. These changes include but not only limited to creation and modification as well as explicit and implicit dependencies of files.

-- Claim 7 – 8, 16 – 17, and 25 – 26: **Willadsen** discloses a system as in Claim 6 above, and he further discloses that the identified external dependencies and generated implicit dependencies of the project software components are stored in memory (i.e.,

Art Unit: 2191

combined dependency table data structure) (Figs. 3 – 4, [0053 – 0055]). In the application under examination, **Whalley** discloses that a combined dependency table data structure is derived from a plurality of dependency tables created during operation of each of the sub-tools of the software project building tool. It is the combined dependency table that is ultimately used to provide information for the system in **Whalley** to perform incremental building of the project; the purpose of the individual dependency tables of each sub-tool is only for providing information in contribution to the formation of the comprised dependency table. Although **Willadsen** does not explicitly mention specific and individual dependency record for each sub-tool is created during its operation, it would have been obvious to one with ordinary skills in the art of software development at the time the invention was made to recognize that specific dependency record for a sub-tool (e.g., syntax analyzer or object linker) must be generated prior to combining and organizing all explicit and implicit dependency information of the entire project into single entity in the system of **Willadsen**.

5. Claims 9, 18, 27, and 28 – 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Willadsen (10/281,945)** and in view of **Hamby (5,848,274)**.

-- Claim 9, 18, and 27: **Willadsen** discloses a system as in Claim 4 above; however, he does not explicitly mention the fact that the project building tool uses the recorded dependency information to determine which of a plurality of build steps need to be executed based on times associated with input and output files. **Hamby** discloses a incremental builder for automatically generating intermediate language of a program

and an incremental imager for dynamically forming the image of the program from the code objects. The incremental builder determines if the source code has been previously compiled and saved in a persistent symbol table; if not, the incremental builder translates the entire source code if the file(s); if yes, the compiler only translates the portions of the source code whose code objects are affected by the edited portions of the original source code. The incremental imager then forms an updated image of the program based of the intermediate language symbols and code objects generated by the incremental builder (**Hamby**, page 6 – 7). In summary, Hamby shows that dependency information of source file(s), intermediate language and code objects are utilized by different tools in his application (i.e. incremental builder and imager) to determine which of the execution steps needs to be carried out. The changes that affect compilation and image formation include time, date, and contents of the source file(s) and its subsequently generated intermediate language symbols and object code. The system disclosed in Claim 4 of **Willadsen** is applicable to different software building projects ([0017]), including compiling source file(s), linking generated object codes, creating executable image, and forming a explicit/implicit dependency relationship graphically (i.e., sub-tools of a software project building tool) (Figs. 3 – 11). It is necessary for the system of Claim 4 of **Willadsen** to determine the sequence of execution of the above-mentioned steps to properly generate dependency information for incremental build as in **Hamby's** disclosure. Thus, it would have been obvious to one with ordinary skills in the art of software development at the time the invention was made to recognize that the system in Claim 4 of **Willadsen** uses the recorded

dependency information to determine the sequence of execution of its build steps as the dependency information (e.g., time, date, and contents) of the source file(s) is updated.

-- Claim 28: **Willadsen** discloses a system for performing software configuration management operations and for supporting build processes, comprising:

a) a processor which is capable of executing all and any software applications or modules (e.g., software building tool) that are specific to the operating system on which the processor runs (Fig. 1, [0046]);

b) a memory for storing software component (e.g., program and data files), management information database (e.g., relationship information, nodal tree structure information, dependencies) and software management modules (i.e., files);

c) a change management monitoring and decision sub-modules which monitor software components to detect changes in the software components as well as explicit and/or implicit dependencies (i.e., monitoring the operation of the data transformation tool) (Figs. 5 – 6, [0058 – 0061]). An initial management information data generator and change management control modules automatically determine explicit and implicit dependencies for the software components (Figs. 3 – 4, [0053 – 0055]);

d) **Willadsen** does not explicitly mention the fact that the software control management tool uses the recorded dependency information to determine which of a plurality of build steps need to be executed based on times associated with input and output files. However, **Hamby** discloses an incremental builder for automatically generating intermediate language of a program and an incremental imager for

dynamically forming the image of the program from the code objects. The incremental builder determines if the source code has been previously compiled and saved in a persistent symbol table; if not, the incremental builder translates the entire source code if the file(s); if yes, the compiler only translates the portions of the source code whose code objects are affected by the edited portions of the original source code. The incremental imager then forms an updated image of the program based of the intermediate language symbols and code objects generated by the incremental builder (**Hamby**, page 6 – 7). In summary, Hamby shows that dependency information of source file(s), intermediate language and code objects are utilized by different tools in his application (i.e. incremental builder and imager) to determine which of the execution steps needs to be carried out. The changes that affect compilation and image formation include time, date, and contents of the source file(s) and its subsequently generated intermediate language symbols and object code. The system disclosed by **Willadsen** is applicable to different software building projects ([0017]), including compiling source file(s), linking generated object codes, creating executable image, and forming a explicit/implicit dependency relationship graphically (i.e., sub-tools of a software project building tool) (Figs. 3 – 11). It is necessary for the system of **Willadsen** to determine the sequence of execution of the above-mentioned steps to properly generate dependency information for incremental build as in **Hamby**'s disclosure. Thus, it would have been obvious to one with ordinary skills in the art of software development at the time the invention was made to recognize that the system of **Willadsen** uses the recorded dependency information to determine the sequence of execution of it build

Art Unit: 2191

steps as the dependency information (e.g., time, date, and contents) of the source file(s) is updated.

-- Claim 29: **Willadsen** and **Hamby** disclose a system as in Claim 28 above, and **Willadsen** further discloses that the identified explicit dependencies (i.e., input/output files and data objects) and generated implicit dependencies of the project software components are stored in memory (i.e., dependency table data structure) (Figs. 3 – 4, [0053 – 0055]).

-- Claims 30 and 31: **Willadsen** and **Hamby** disclose a system as in Claim 28 above, and **Willadsen** further discloses that the monitoring, decision, and software component action modules monitor changes of explicit and implicit dependencies (e.g., input, intermediate, and output files). These modules can be manually executed by the user or automatically invoked by the software management components themselves when the software control management tool detects changes in dependencies (Figs. 5 – 6, [0058 – 0061]).

-- Claim 32: **Willadsen** and **Hamby** disclose a system as in Claim 28 above, and **Willadsen** further discloses that the software management components can automatically invoke the monitoring, decision, and software component action modules to monitor and record changes of explicit and implicit dependencies (i.e., on the fly) (Figs. 5 – 6, [0058 – 0061]).

-- Claims 33 and 34: **Willadsen** and **Hamby** disclose a system as in Claim 28 above; however, neither reference explicitly mentions the specific dependency monitoring technique employed in his invention. In the application under examination, The Applicant discloses TSD (referring to a specific approach for running low-level driver-type software in DOS for hooking all file access in a system), FSFD (referring to a specific type of driver in Windows NT/2000/XP for hooking all file access in a system), and LD_LIBRARY_PRELOAD (referring to a specific technique approach on Unix-type OS for hooking all file access from a given program) as possible methods of file access monitoring (pages 14 – 15). Additionally, The Applicant also cites while TSD/FSFD and LD_LIBRARY_PRELOAD file system monitoring are conventional approaches (page 14), his claimed invention is not limited to them and various approaches to performing the required monitoring can be used (page 16). Since The Applicant does not specify any critical or significant advantages for using TSD/FSFD or LD_LIBRARY_LOAD file system monitoring over other known techniques in the system of Claim 28, the fact of merely applying one of these conventional file system monitoring techniques in the application under examination would have been obvious to one with ordinary skills in the art of software development at the time the invention was made.

-- Claims 35 and 36: **Willadsen** and **Hamby** disclose a system as in Claim 28 above, and **Willadsen** further discloses that the Nodal Tree Structure which is indicative and illustrative of the explicit/implicit dependencies and their updated changes can be

Art Unit: 2191

displayed to the user through a network or communication interface in the system (Fig. 1, [0054]).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Thai Van Pham whose telephone number is (571) 270-1064. The examiner can normally be reached on Monday - Thursday, 9am - 5pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, James Myhre can be reached on (571) 270-1065. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TVP
TVP


James Myhre
Supervisory Patent Examiner